Signatures from Short Basis Lattice Trapdoors

Edward Eaton

University of Waterloo

December 4, 2015

Overview

Say we have function f, trapdoor information providing f^{-1}.

Say we have function f, trapdoor information providing f^{-1}.
We can construct a Signature scheme as follows:

Say we have function f, trapdoor information providing f^{-1}.
We can construct a Signature scheme as follows:

- G: public key is f and hash function H, secret key is f^{-1}

Say we have function f, trapdoor information providing f^{-1}.
We can construct a Signature scheme as follows:

- G: public key is f and hash function H, secret key is f^{-1}
- S: Given a message m, compute $\sigma \leftarrow f^{-1}(H(m))$

Say we have function f, trapdoor information providing f^{-1}.
We can construct a Signature scheme as follows:

- G: public key is f and hash function H, secret key is f^{-1}
- S: Given a message m, compute $\sigma \leftarrow f^{-1}(H(m))$
- V: Check that $H(m) = f(\sigma)$

Say we have function f, trapdoor information providing f^{-1}.
We can construct a Signature scheme as follows:

- G: public key is f and hash function H, secret key is f^{-1}
- S: Given a message m, compute $\sigma \leftarrow f^{-1}(H(m))$
- V: Check that $H(m) = f(\sigma)$

Scheme is EU-CMA in ROM assuming f is hard to invert (with some assumptions on f).

For those who dislike ROM:

For those who dislike ROM:

- G: public key is $\{f_m : m \in \mathcal{M}\}$ and public parameter s, secret key is $\{f_m^{-1} : m \in \mathcal{M}\}$

For those who dislike ROM:

- G: public key is $\{f_m : m \in \mathcal{M}\}$ and public parameter s, secret key is $\{f_m^{-1} : m \in \mathcal{M}\}$
  That is, associate with each message m a public function f_m for which you control trapdoor information

For those who dislike ROM:

- G: public key is $\{f_m : m \in \mathcal{M}\}$ and public parameter s, secret key is $\{f_m^{-1} : m \in \mathcal{M}\}$
  That is, associate with each message m a public function f_m for which you control trapdoor information
- S: Given a message m, compute $\sigma \leftarrow f_m^{-1}(s)$

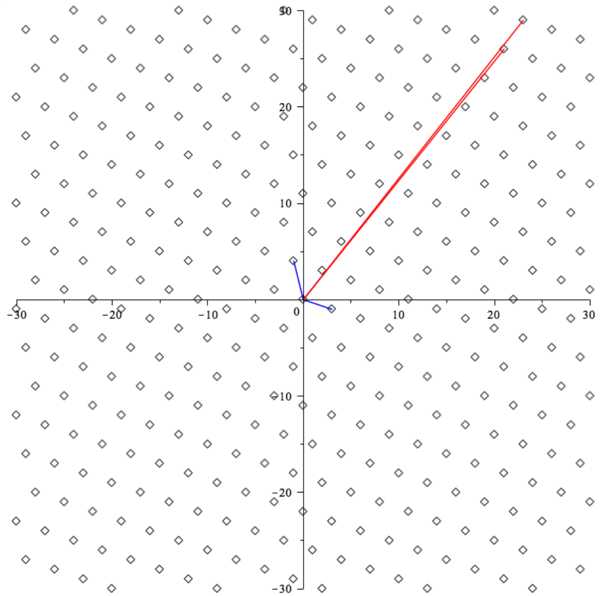
For those who dislike ROM:

- G: public key is $\{f_m : m \in \mathcal{M}\}$ and public parameter s, secret key is $\{f_m^{-1} : m \in \mathcal{M}\}$
  That is, associate with each message m a public function f_m for which you control trapdoor information
- S: Given a message m, compute $\sigma \leftarrow f_m^{-1}(s)$
- V: Check that $f_m(s) = \sigma$

Quality of output of lattice algorithms is generally related to $||\tilde{b}_i||$ (Graham-Schmidt orthogonalization of basis vectors).

Quality of output of lattice algorithms is generally related to $||\tilde{b}_i||$ (Graham-Schmidt orthogonalization of basis vectors).

Lattices admit multiple bases. Easy to get a 'bad' basis from a 'good' basis, hard to do reverse.


30
20
10
0
−10
−20
−30
−30
−20
−10
10
20
30

Generate a good basis S and a bad basis B.
The function f will depend on the lattice generated by B.
The inverse f^{-1} will depend on solving a problem the nice basis S allows.

We need a methodology to generate a short basis S along with a hard basis B.

We would like this method to satisfy:

We need a methodology to generate a short basis S along with a hard basis B.

We would like this method to satisfy:

- Fast to do

We need a methodology to generate a short basis S along with a hard basis B.

We would like this method to satisfy:

- Fast to do
- B reveals no information about S other than $\mathcal{L}(B) = \mathcal{L}(S)$

We need a methodology to generate a short basis S along with a hard basis B.
We would like this method to satisfy:

- Fast to do
- B reveals no information about S other than $\mathcal{L}(B) = \mathcal{L}(S)$
- S is high quality - vectors are very short and relatively orthogonal

We need a methodology to generate a short basis S along with a hard basis B.

We would like this method to satisfy:

- Fast to do
- B reveals no information about S other than $\mathcal{L}(B) = \mathcal{L}(S)$
- S is high quality - vectors are very short and relatively orthogonal
- $\mathcal{L}(B)$ has an appropriate distribution for average-case to worst-case reduction

Definitions

For a matrix $A \in \mathbb{Z}_q^{n\times m}$, the lattice associated with A is

$$\Lambda^{\perp}(A) := \left\{x \in \mathbb{Z}^m : Ax = 0 \in \mathbb{Z}_q^n\right\}$$

For a basis S, $\tilde{S}$ denotes the Graham-Schmidt orthogonalization of S. $||\tilde{S}|| = \max_i ||\tilde{s}_i||$, (the norm of the basis is the norm of the largest vector)

For a lattice Λ, the discrete gaussian centered at c with parameter s, $D_{\Lambda,c,s}$, is the distribution where for all $x \in \Lambda$, the probability of selecting x is proportional to

$$\exp(-\pi||x-c||^2/s^2)$$

GenBasis Algorithm

Alwen, Peikert (2010):
There is a fixed constant $C > 1$ and a probabilistic polynomial-time algorithm $GenBasis(1^n, 1^m, q)$ that, for $poly(n)$-bounded $m \geq Cn \log q$ outputs $A \in \mathbb{Z}_q^{n \times m}$ and $S \in \mathbb{Z}^{m \times m}$ such that:

- The distribution of the output A is negligibly (in n) close to uniform
- S is a basis of $\Lambda^{\perp}(A)$
- $||S|| \leq O(\sqrt{n \log q})$

GenBasis Algorithm

Take in parameters n, m, q. Output matrix $A \in \mathbb{Z}_q^{n \times m}$ negligibly close to uniform and basis S of $\Lambda^{\perp}(A)$.

GenBasis Algorithm

Take in parameters n, m, q. Output matrix $A \in \mathbb{Z}_q^{n \times m}$ negligibly close to uniform and basis S of $\Lambda^{\perp}(A)$.
General idea: Let $m = m_1 + m_2$. Generate $A_1 \in \mathbb{Z}_q^{n \times m_1}$ uniformly at random. We will construct the other 'half' of the matrix A_2 to get $A = A_1 || A_2$ at the same time as a basis S.

GenBasis Algorithm

Generate:

- $U \in \mathbb{Z}^{m_1 \times m_2}$, non singular
- $R \in \mathbb{Z}^{m_1 \times m_2}$, random 'short' matrix
- $G \in \mathbb{Z}^{m_1 \times m_2}$, with entries increasing left to right geometrically
- $P \in \mathbb{Z}^{m_2 \times m_1}$ picking out certain columns of G via GP
- $C \in \mathbb{Z}^{m_1 \times m_1}$ such that $GP + C \subset \Lambda^{\perp}(A_1)$

Set $A_2 = -A_1 \cdot (R + G) \in \mathbb{Z}_q^{n \times m_2}$. Set $A = A_1 || A_2$.
Then

$$S = \begin{pmatrix} (G+R)U & RP - C \\ U & P \end{pmatrix}$$

So we can generate pairs (A, S), where S is a good basis for $\Lambda^{\perp}(A)$, can't get S from A.

So we can generate pairs (A, S), where S is a good basis for $\Lambda^{\perp}(A)$, can't get S from A.
Need to turn this into functions f, f^{-1}, where f depends on A, f^{-1} depends on A and S.

Function Definition

$$f_A : \{ \text{ Short vectors in } \mathbb{Z}^m \} \to \mathbb{Z}^n$$

Function Definition

$$f_A : \{ \text{ Short vectors in } \mathbb{Z}^m \} \to \mathbb{Z}^n$$

$$f_A : x \mapsto Ax$$

Function Definition

$$f_A : \{\text{ Short vectors in } \mathbb{Z}^m\} \to \mathbb{Z}^n$$

$$f_A : x \mapsto Ax$$

Why is f hard to invert?

- Inhomogenous small integer solution problem (ISIS):

- Inhomogenous small integer solution problem (ISIS): Given integer q, matrix $A \in \mathbb{Z}_q^{n \times m}$, syndrome $u \in \mathbb{Z}_q^n$, real number β find integer vector $e \in \mathbb{Z}^m$ such that $Ae = u \pmod{q}$ and $||e||_2 \leq \beta$.

- Inhomogenous small integer solution problem (ISIS): Given integer q, matrix $A \in \mathbb{Z}_q^{n \times m}$, syndrome $u \in \mathbb{Z}_q^n$, real number β find integer vector $e \in \mathbb{Z}^m$ such that $Ae = u \pmod{q}$ and $||e||_2 \leq \beta$.
- Small integer solution problem (SIS): ISIS with $u = 0$

GPV08:
For any poly-bounded m, $\beta = poly(n)$ and for any prime $q \geq \beta \cdot \omega(\sqrt{n \log n})$, the average-case problems $\mathrm{SIS}_{q,m,\beta}$ and $\mathrm{ISIS}_{q,m,\beta}$ are as hard as approximating the SIVP (Shortest independent vectors problem) (among other problems) in the worst case to within certain $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$ factors.

GPV08:
For any poly-bounded m, $\beta = poly(n)$ and for any prime $q \geq \beta \cdot \omega(\sqrt{n \log n})$, the average-case problems $\mathrm{SIS}_{q,m,\beta}$ and $\mathrm{ISIS}_{q,m,\beta}$ are as hard as approximating the SIVP (Shortest independent vectors problem) (among other problems) in the worst case to within certain $\gamma = \beta \cdot \tilde{O}(\sqrt{n})$ factors.
So inverting f_A (without S) should be hard.

SampleD

GPV08:
There is a probabilistic polynomial-time algorithm (SampleD) that, given a basis S of an n-dimensional lattice $\Lambda = \mathcal{L}(S)$, a parameter $s \geq ||\tilde{S}|| \cdot \omega(\sqrt{\log n})$, and a center $c \in \mathbb{R}^n$, outputs a sample from a distribution that is statistically close to $D_{\Lambda,s,c}$

SampleD

GPV08:
There is a probabilistic polynomial-time algorithm (SampleD) that, given a basis S of an n-dimensional lattice $\Lambda = \mathcal{L}(S)$, a parameter $s \geq ||\tilde{S}|| \cdot \omega(\sqrt{\log n})$, and a center $c \in \mathbb{R}^n$, outputs a sample from a distribution that is statistically close to $D_{\Lambda,s,c}$
Crucial Note: Distribution of output does not depend on S - we reveal no information about basis (unlike GGH, NTRUSign, etc.)

Sample$\mathbb{Z}$

Sample$\mathbb{Z}$ samples from the discrete Gaussian $D_{\mathbb{Z},s,c}$. It works as follows:

Sample$\mathbb{Z}$

Sample$\mathbb{Z}$ samples from the discrete Gaussian $D_{\mathbb{Z},s,c}$. It works as follows:

Define some function $t(n) \geq \omega(\sqrt{\log n})$ (e.g. $t(n) = \log n$)

Sample$\mathbb{Z}$

Sample$\mathbb{Z}$ samples from the discrete Gaussian $D_{\mathbb{Z},s,c}$. It works as follows:
Define some function $t(n) \geq \omega(\sqrt{\log n})$ (e.g. $t(n) = \log n$)
Sample $x \leftarrow [c - s \cdot t(n), c + s \cdot t(n)] \cap \mathbb{Z}$ uniformly

Sample$\mathbb{Z}$

Sample$\mathbb{Z}$ samples from the discrete Gaussian $D_{\mathbb{Z},s,c}$. It works as follows:
Define some function $t(n) \geq \omega(\sqrt{\log n})$ (e.g. $t(n) = \log n$)
Sample $x \leftarrow [c - s \cdot t(n), c + s \cdot t(n)] \cap \mathbb{Z}$ uniformly
With probability $\rho_s(x - c) = \exp(\pi^2|x - c|^2/s^2)$, output x.
Otherwise, repeat.

SampleD

The SampleD algorithm samples from the discrete Gaussian Λ. It takes as input a basis for a lattice B, a Gaussian parameter s, a centre $c \in \mathbb{R}^n$.

SampleD

The SampleD algorithm samples from the discrete Gaussian Λ. It takes as input a basis for a lattice B, a Gaussian parameter s, a centre $c \in \mathbb{R}^n$.

1. Let $v_0 = 0$ and $c_0 = c$.

SampleD

The SampleD algorithm samples from the discrete Gaussian Λ. It takes as input a basis for a lattice B, a Gaussian parameter s, a centre $c \in \mathbb{R}^n$.

1. Let $v_0 = 0$ and $c_0 = c$.
2. For i from 0 to $n - 1$:

SampleD

The SampleD algorithm samples from the discrete Gaussian Λ. It takes as input a basis for a lattice B, a Gaussian parameter s, a centre $c \in \mathbb{R}^n$.

1. Let $v_0 = 0$ and $c_0 = c$.
2. For i from 0 to $n-1$:
    1. Let $c_i' = \frac{\langle c_i, \tilde{b}_i \rangle}{\langle \tilde{b}_i, \tilde{b}_i \rangle}$ and $s_i' = \frac{s}{||\tilde{b}_i||}$

SampleD

The SampleD algorithm samples from the discrete Gaussian Λ. It takes as input a basis for a lattice B, a Gaussian parameter s, a centre $c \in \mathbb{R}^n$.

1. Let $v_0 = 0$ and $c_0 = c$.
2. For i from 0 to $n-1$:
    1. Let $c_i' = \frac{\langle c_i, \tilde{b}_i \rangle}{\langle \tilde{b}_i, \tilde{b}_i \rangle}$ and $s_i' = \frac{s}{||\tilde{b}_i||}$
    2. Choose $z_i \leftarrow D_{\mathbb{Z}, s_i', c_i'}$

SampleD

The SampleD algorithm samples from the discrete Gaussian Λ. It takes as input a basis for a lattice B, a Gaussian parameter s, a centre $c \in \mathbb{R}^n$.

1. Let $v_0 = 0$ and $c_0 = c$.
2. For i from 0 to $n-1$:
    1. Let $c_i' = \frac{\langle c_i, \tilde{b}_i \rangle}{\langle \tilde{b}_i, \tilde{b}_i \rangle}$ and $s_i' = \frac{s}{||\tilde{b}_i||}$
    2. Choose $z_i \leftarrow D_{\mathbb{Z}, s_i', c_i'}$
    3. $c_{i+1} = c_i - z_i b_i$, $v_{i+1} = v_i + z_i b_i$

SampleD

The SampleD algorithm samples from the discrete Gaussian Λ. It takes as input a basis for a lattice B, a Gaussian parameter s, a centre $c \in \mathbb{R}^n$.

1. Let $v_0 = 0$ and $c_0 = c$.
2. For i from 0 to $n-1$:
   1. Let $c_i' = \frac{\langle c_i, \tilde{b}_i \rangle}{\langle \tilde{b}_i, \tilde{b}_i \rangle}$ and $s_i' = \frac{s}{||\tilde{b}_i||}$
   2. Choose $z_i \leftarrow D_{\mathbb{Z}, s_i', c_i'}$
   3. $c_{i+1} = c_i - z_i b_i$, $v_{i+1} = v_i + z_i b_i$
3. Output v_n

Can construct f_A^{-1} as follows:

Can construct f_A^{-1} as follows:
SampleISIS takes in a matrix A, a short basis S for $\Lambda^{\perp}(A)$, a gaussian parameter $s \geq ||\tilde{S}|| \cdot \omega(\sqrt{\log m})$, and $u \in \mathbb{Z}^n$. It outputs $e \in \mathbb{Z}^m$ such that $||e||_2 \leq s\sqrt{m}$ and $Ae = u$.

Can construct f_A^{-1} as follows:
SampleISIS takes in a matrix A, a short basis S for $\Lambda^{\perp}(A)$, a gaussian parameter $s \geq ||\tilde{S}|| \cdot \omega(\sqrt{\log m})$, and $u \in \mathbb{Z}^n$. It outputs $e \in \mathbb{Z}^m$ such that $||e||_2 \leq s\sqrt{m}$ and $Ae = u$.

- Choose arbitrary $t \in \mathbb{Z}^m$ such that $At = u \pmod{q}$ (find with linear algebra. t need not be short).

Can construct f_A^{-1} as follows:
SampleISIS takes in a matrix A, a short basis S for $\Lambda^{\perp}(A)$, a gaussian parameter $s \geq ||\tilde{S}|| \cdot \omega(\sqrt{\log m})$, and $u \in \mathbb{Z}^n$. It outputs $e \in \mathbb{Z}^m$ such that $||e||_2 \leq s\sqrt{m}$ and $Ae = u$.

- Choose arbitrary $t \in \mathbb{Z}^m$ such that $At = u \pmod{q}$ (find with linear algebra. t need not be short).
- Sample $v \leftarrow \text{SampleD}(S, s, -t)$

Can construct f_A^{-1} as follows:
SampleISIS takes in a matrix A, a short basis S for $\Lambda^{\perp}(A)$, a gaussian parameter $s \geq ||\tilde{S}|| \cdot \omega(\sqrt{\log m})$, and $u \in \mathbb{Z}^n$. It outputs $e \in \mathbb{Z}^m$ such that $||e||_2 \leq s\sqrt{m}$ and $Ae = u$.

- Choose arbitrary $t \in \mathbb{Z}^m$ such that $At = u \pmod q$ (find with linear algebra. t need not be short).
- Sample $v \leftarrow \text{SampleD}(S, s, -t)$
- Output $e = t + v$

Then $Ae = At + Av = u + 0 = u$ and e is short.

Probabilistic Full Domain Hash (PFDH) From GPV08:

Parameters: Security parameter n, modulus q, dimension $m = O(n \log q)$, bound $\beta = O(\sqrt{m})$, salt length k

Probabilistic Full Domain Hash (PFDH) From GPV08:

Parameters: Security parameter n, modulus q, dimension $m = O(n \log q)$, bound $\beta = O(\sqrt{m})$, salt length k

- $Gen(1^n)$: $(A, S) \leftarrow GenBasis(1^n, 1^m, q)$ Public key is A, private key is S.

Probabilistic Full Domain Hash (PFDH) From GPV08:

Parameters: Security parameter n, modulus q, dimension $m = O(n \log q)$, bound $\beta = O(\sqrt{m})$, salt length k

- $Gen(1^n)$: $(A, S) \leftarrow GenBasis(1^n, 1^m, q)$ Public key is A, private key is S.
- $Sig(S, msg)$: Choose $r \xleftarrow{\$} \{0,1\}^k$. Compute $u = H(msg||r)$. $e \leftarrow \text{SampleISIS}(A, S, ||\tilde{S}||\beta, u)$. $\sigma = (e, r)$

Probabilistic Full Domain Hash (PFDH) From GPV08:

Parameters: Security parameter n, modulus q, dimension $m = O(n \log q)$, bound $\beta = O(\sqrt{m})$, salt length k

- $Gen(1^n)$: $(A, S) \leftarrow GenBasis(1^n, 1^m, q)$ Public key is A, private key is S.
- $Sig(S, msg)$: Choose $r \xleftarrow{\$} \{0,1\}^k$. Compute $u = H(msg||r)$. $e \leftarrow \text{SampleISIS}(A, S, ||\tilde{S}||\beta, u)$. $\sigma = (e, r)$
- $Ver(A, msg, \sigma = (e, r))$: Check that $Ae = H(msg||r)$ and that $||e||_2 \leq \beta\sqrt{m}$. If so, accept. Otherwise, reject.

Security Reduction

Given A, find a short e such that $Ae = 0$

Security Reduction

Given A, find a short e such that $Ae = 0$
On hash query $(msg_i||r_i)$:

Security Reduction

Given A, find a short e such that $Ae = 0$

On hash query $(msg_i || r_i)$:

- Choose $e_i \in \mathbb{Z}_q^m$

Security Reduction

Given A, find a short e such that $Ae = 0$

On hash query $(msg_i||r_i)$:

- Choose $e_i \in \mathbb{Z}_q^m$
- Set $H(msg||r) = Ae_i$

Security Reduction

Given A, find a short e such that $Ae = 0$

On hash query $(msg_i || r_i)$:

- Choose $e_i \in \mathbb{Z}_q^m$
- Set $H(msg||r) = Ae_i$

On signing query msg:

Security Reduction

Given A, find a short e such that $Ae = 0$

On hash query $(msg_i||r_i)$:

- Choose $e_i \in \mathbb{Z}_q^m$
- Set $H(msg||r) = Ae_i$

On signing query msg:

- Choose $r \in \{0,1\}^k$

Security Reduction

Given A, find a short e such that $Ae = 0$

On hash query $(msg_i||r_i)$:

- Choose $e_i \in \mathbb{Z}_q^m$
- Set $H(msg||r) = Ae_i$

On signing query msg:

- Choose $r \in \{0,1\}^k$
- Find e corresponding to $(msg||r)$ in hash table

Security Reduction

Given A, find a short e such that $Ae = 0$

On hash query $(msg_i||r_i)$:

- Choose $e_i \in \mathbb{Z}_q^m$
- Set $H(msg||r) = Ae_i$

On signing query msg:

- Choose $r \in \{0,1\}^k$
- Find e corresponding to $(msg||r)$ in hash table
- Output (e, r).

When forgery msg^*, e^*, r^* is received, look up $msg^*||r^*$ in hash table and find corresponding e.

When forgery msg^*, e^*, r^* is received, look up $msg^* || r^*$ in hash table and find corresponding e.
With high probability, $e \neq e^*$ (recall that $\mathcal{A}$ never asked for a signature on msg^*).

When forgery msg^*, e^*, r^* is received, look up $msg^*||r^*$ in hash table and find corresponding e.
With high probability, $e \neq e^*$ (recall that $\mathcal{A}$ never asked for a signature on msg^*).
So $A(e - e^*) = 0$, and we have broken SIS resistance of A.

Bonsai Trees

Given a basis S for $\Lambda^{\perp}(A)$, note that we can generate a (similarly nice) basis for $\Lambda^{\perp}(A||C)$ for any matrix C.

Bonsai Trees

Given a basis S for $\Lambda^{\perp}(A)$, note that we can generate a (similarly nice) basis for $\Lambda^{\perp}(A||C)$ for any matrix C.
Let W be (any) solution to $AW = -C$. Then let

$$S' = \begin{pmatrix} S & W \\ 0 & I \end{pmatrix}$$

Bonsai Trees

Given a basis S for $\Lambda^{\perp}(A)$, note that we can generate a (similarly nice) basis for $\Lambda^{\perp}(A||C)$ for any matrix C.
Let W be (any) solution to $AW = -C$. Then let

$$S' = \begin{pmatrix} S & W \\ 0 & I \end{pmatrix}$$

Then $(A||C)S' = AS + AW + C = 0 + -C + C = 0$

As well, $\tilde{S}' = \begin{pmatrix} S & 0 \\ 0 & I \end{pmatrix}$ so $||\tilde{S}'|| = ||\tilde{S}||$

As well, $\tilde{S}' = \begin{pmatrix} S & 0 \\ 0 & I \end{pmatrix}$ so $||\tilde{S}'|| = ||\tilde{S}||$

And for any $v \in \Lambda^{\perp}(A||C)$ write $v = v_1||v_2$ Then

$$0 = (A||C)(v_1||v_2) = Av_1 + Cv_2 = Av_1 - (AW)v_2 = A(v_1 - Wv_2)$$

As well, $\tilde{S}' = \begin{pmatrix} S & 0 \\ 0 & I \end{pmatrix}$ so $||\tilde{S}'|| = ||\tilde{S}||$

And for any $v \in \Lambda^{\perp}(A||C)$ write $v = v_1||v_2$ Then

$$0 = (A||C)(v_1||v_2) = Av_1 + Cv_2 = Av_1 - (AW)v_2 = A(v_1 - Wv_2)$$

So let e_1 be such that $Se_1 = v_1 - Wv_2$. Then let $e = e_1||v_2$.

$$S'e = S'(e_1||v_2) = (Se_1 + Wv_2)||v_2 = (v_1 - Wv_2 + Wv_2)||v_2 = v_1||v_2 = v$$

Bonsai Tree Signature Scheme

Gen(1^n): $(A_0, S_0) \leftarrow GenBasis(1^n, 1^m, q)$. For $i \in \{1, \ldots, k\}$ and $b \in \{0, 1\}$, generate $A_i^{(b)} \in \mathbb{Z}_q^{n \times m}$ uniformly.

Public key is $A_0, \{A_i^{(b)}\}$. Secret key is S_0.

$Sig(S_0, msg)$: Let $\mu = H(msg) \in \{0,1\}^k$. Define the matrix

$$A_\mu = A_0||A_1^{(\mu_1)}||A_2^{(\mu_2)}||\ldots||A_k^{(\mu_k)}$$

Let $S_\mu \leftarrow \text{ExtBasis}(S_0, A_0, A_\mu)$.
Then take $\sigma \leftarrow \text{SampleISIS}(A_\mu, S_\mu, ||\tilde{S}||\beta, 0)$.
Signature is σ

$Ver(A_0, \{A_i^{(j)}\}, \sigma, msg)$:
Construct A_μ as above. Accept if σ is a short vector in $\Lambda^\perp(A_\mu)$

	Security	Model	$\lvert pk \rvert$	$\lvert sk \rvert$	$\lvert sig \rvert$
PFDH	su-acma	R.O.M.	nm	m^2	$m + \lvert r \rvert$
Bonsai	eu-scma	Standard	$(2k+1)nm$	m^2	$(k+1)m$

Boyen, 2010

Rather than having $A_0, \{A_i^{(j)}\}$ and setting
$A_\mu = A_0 || A_1^{(\mu_1)} || \ldots || A_k^{(\mu_k)}$
Instead have $A_0, \{A_i\}$ and set

$$A_\mu = A_0 + \sum_{i=1}^{k} (-1)^{\mu_1} A_i$$

Can still create a new basis using S.

	Security	Model	$\lvert pk \rvert$	$\lvert sk \rvert$	$\lvert sig \rvert$
PFDH	su-acma	R.O.M.	nm	m^2	$m + \lvert r \rvert$
Bonsai	eu-scma	Standard	$(2k+1)nm$	m^2	$(k+1)m$
Boyen	eu-acma	Standard	$(k+1)nm$	m^2	m

Rückert, 2010

Strong Unforgeability: A forgery (m^*, σ^*) is considered valid if m^* was never queried *or* σ^* was not the response when m^* was queried.

Rückert, 2010

Strong Unforgeability: A forgery (m^*, σ^*) is considered valid if m^* was never queried *or* σ^* was not the response when m^* was queried.

Bonsai trees not strongly unforgeable - If σ is a signature, so is $-\sigma$

Rückert, 2010

Strong Unforgeability: A forgery (m^*, σ^*) is considered valid if m^* was never queried *or* σ^* was not the response when m^* was queried.

Bonsai trees not strongly unforgeable - If σ is a signature, so is $-\sigma$

Rather than sampling preimages to the zero vector, sample preimages to a vector u, which is part of the public key.

	Security	Model	$\|pk\|$	$\|sk\|$	$\|sig\|$
PFDH	su-acma	R.O.M.	nm	m^2	$m + \|r\|$
Bonsai	eu-scma	Standard	$(2k+1)nm$	m^2	$(k+1)m$
Boyen	eu-acma	Standard	$(k+1)nm$	m^2	m
Rückert	su-scma	Standard	$(2k+1)nm + m$	m^2	$(k+1)m$

Boneh & Zhandry, 2013

Noted that GPV08 Signature scheme security was shown in R.O.M., not Q.R.O.M.
Reproved security in Q.R.O.M. (in fact, showed quantum existential unforgeability under chosen message attack)

	Security	Model	$\|pk\|$	$\|sk\|$	$\|sig\|$
PFDH	q-eu-acma	Q.R.O.M.	nm	m^2	$m + \|r\|$
Bonsai	eu-scma	Standard	$(2k+1)nm$	m^2	$(k+1)m$
Boyen	eu-acma	Standard	$(k+1)nm$	m^2	m
Rückert	su-scma	Standard	$(2k+1)nm + m$	m^2	$(k+1)m$

Teranishi, Oyama, Ogata (2006): There is a generic conversion from eu-acma signature schemes to su-acma signature schemes based on the collision resistance of a Chameleon Hash function.

Teranishi, Oyama, Ogata (2006): There is a generic conversion from eu-acma signature schemes to su-acma signature schemes based on the collision resistance of a Chameleon Hash function.
Chameleon Hash function: a function f that can be generated with trapdoor information td such that without td, f is collision resistant, but with td, f is preimage-sampleable.

Teranishi, Oyama, Ogata (2006): There is a generic conversion from eu-acma signature schemes to su-acma signature schemes based on the collision resistance of a Chameleon Hash function.
Chameleon Hash function: a function f that can be generated with trapdoor information td such that without td, f is collision resistant, but with td, f is preimage-sampleable.
Note - Our f_A with secret basis S is just this!

Teranishi, Oyama, Ogata (2006): There is a generic conversion from eu-acma signature schemes to su-acma signature schemes based on the collision resistance of a Chameleon Hash function.

Chameleon Hash function: a function f that can be generated with trapdoor information td such that without td, f is collision resistant, but with td, f is preimage-sampleable.

Note - Our f_A with secret basis S is just this!

Teranishi, Oyama, Ogata proved security of transformation in R.O.M. with respect to a specific (discrete-log based) implementation of a chameleon hash.

Teranishi, Oyama, Ogata (2006): There is a generic conversion from eu-acma signature schemes to su-acma signature schemes based on the collision resistance of a Chameleon Hash function.

Chameleon Hash function: a function f that can be generated with trapdoor information td such that without td, f is collision resistant, but with td, f is preimage-sampleable.

Note - Our f_A with secret basis S is just this!

Teranishi, Oyama, Ogata proved security of transformation in R.O.M. with respect to a specific (discrete-log based) implementation of a chameleon hash.

Eaton and Song proved security of transformation is Q.R.O.M. with respect to a generic chameleon hash.

	Security	Model	$\lvert pk \rvert$	$\lvert sk \rvert$	$\lvert sig \rvert$
PFDH	q-eu-acma	Q.R.O.M.	nm	m^2	$m + \lvert r \rvert$
Bonsai	eu-scma	Standard	$(2k+1)nm$	m^2	$(k+1)m$
Boyen	eu-acma	Standard	$(k+1)nm$	m^2	m
Rückert	su-scma	Standard	$(2k+1)nm + m$	m^2	$(k+1)m$
Boyen + ES	su-acma	Q.R.O.M.	$(k+1)nm + m^2$	$2m^2$	$2m$

Thank You

Sources:

- *How to Use a Short Basis: Trapdoors for Hard Lattices and New Cryptographic Constructions* by Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan
- *Generating Shorter Bases for Hard Random Lattices* by Joël Alwen and Chris Peikert
- *Bonsai Trees, or How to Delegate a Lattice Basis* by David Cash, Dennis Hofheinz, Eikie Kiltz, Chris Peikert
- *Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and more* by Xavier Boyen
- *Strongly Unforgeable Signatures and Hierarchical Identity-based Signatures from Lattices without Random Oracles* by Markus Rückert
- *Secure Signatures and Chosen Ciphertext Security in a Quantum Computing World* by Dan Boneh and Mark Zhandry
- *General Conversion for Obtaining Strongly Existentially Unforgeable Signatures* by Isamu Teranishi, Takuro Oyama, and Wakaha Ogata
- *Making Existential-Unforgeable Signatures Strongly Unforgeable in the Quantum Random-Oracle Model* by Edward Eaton and Fang Song